3.1 Using Data Types

Data Types

Data type. Set of values and operations on those values.

Primitive types. Values directly map to machine representation; ops directly map to machine instructions.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Set of Values</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true, false</td>
<td>not, and, or, xor</td>
</tr>
<tr>
<td>int</td>
<td>-2^31 to 2^31-1</td>
<td>add, subtract, multiply</td>
</tr>
<tr>
<td>double</td>
<td>any of 2^64 possible reals</td>
<td>add, subtract, multiply</td>
</tr>
</tbody>
</table>

We want to write programs that process other types of data.
- Colors, pictures, strings, input streams, ...
- Complex numbers, vectors, matrices, polynomials, ...
- Points, polygons, charged particles, celestial bodies, ...

Objects

Object. Holds a data type value; variable name refers to object.

Object-oriented programming.
- Create your own data types (set of values and ops on them).
- Use them in your programs (manipulate objects that hold values).

Constructors and Methods

To construct a new object:
- Use keyword `new` (to invoke constructor).
- Use name of data type (to specify which type of object).

To apply an operation:
- Use name of object (to specify which object).
- Use the dot operator (to invoke method).
- Use the name of the method (to specify which operation).

Declare a variable (object name):
```
String s;
```

Call a constructor to create an object:
```
s = new String("Hello, world!");
```

Call a method that operates on the object's value:
```
System.out.println(s.substring(0, 5));
```
**Color Data Type**

**Color.** A sensation in the eye from electromagnetic radiation.

**Set of values.** [RGB representation] 256^3 possible values, which quantify the amount of red, green, and blue, each on a scale of 0 to 255.

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>0</td>
<td>Green</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>0</td>
<td>Blue</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
<td>Cyan</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td>Magenta</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>255</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

**API.** Application Programming Interface.

**Color(int r, int g, int b)**  
- `getRed()`: red intensity  
- `getGreen()`: green intensity  
- `getBlue()`: blue intensity  
- `getBrightness()`: brighter version of this color  
- `darker()`: darker version of this color  
- `toString()`: string representation of the color  
- `equals(Color c)`: is this color's value the same as C?

[http://download.oracle.com/javase/6/docs/api/java/awt/Color.html](http://download.oracle.com/javase/6/docs/api/java/awt/Color.html)

**Monochrome Luminance**

**Monochrome luminance.** Effective brightness of a color.

**NTSC formula.** \( Y = 0.299R + 0.587G + 0.114B \).

```java
import java.awt.Color;
public class Luminance {
    public static double lum(Color c) {
        int r = c.getRed();
        int g = c.getGreen();
        int b = c.getBlue();
        return 0.299 * r + 0.587 * g + 0.114 * b;
    }
}
```

**Color Compatibility**

**Q.** Which font colors will be most readable with which background colors on computer and cell phone screens?

**A.** Rule of thumb: difference in luminance should be \( \geq 128 \).

```java
public static boolean compatible(Color a, Color b) {
    return Math.abs(lum(a) - lum(b)) >= 128.0;
}
```

**Grayscale**

**Grayscale.** When all three R, G, and B values are the same, resulting color is on grayscale from 0 (black) to 255 (white).

**Convert to grayscale.** Use luminance to determine value.

```java
public static Color toGray(Color c) {
    int y = (int) Math.round(lum(c));
    Color gray = new Color(y, y, y);
    return gray;
}
```

**Bottom line.** We are writing programs that manipulate color.

**OOP Context for Color**

**Possible memory representation.**

<table>
<thead>
<tr>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>This color</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>74</td>
<td>74</td>
<td>74</td>
<td>cyan</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Blue</td>
</tr>
</tbody>
</table>

**Object reference is analogous to variable name.**

- We can manipulate the value that it holds.
- We can pass it to (or return it from) a method.
References

René Magritte. "This is not a pipe."

Java. This is not a color.

Color sienna = new Color(160, 82, 45);
Color c = sienna.darker();

OOP. Natural vehicle for studying abstract models of the real world.

This is Not a Pipe

Neither is this.

Image Processing: Grayscale Filter

Goal. Convert color image to grayscale according to luminance formula.

import java.awt.Color;

public class Grayscale {
    public static void main(String[] args) {
        Picture pic = new Picture(args[0]);
        for (int y = 0; y < pic.height(); y++) {
            for (int x = 0; x < pic.width(); x++) {
                Color color = pic.getPixel(x, y);
                Color gray = Luminance.toGray(color);
                pic.set(x, y, gray);
            }
        }
        pic.show();
    }
}

Image Processing: Scaling Filter

Goal. Shrink or enlarge an image to desired size.

Downscaling. To shrink, delete half the rows and columns.
Upscaling. To enlarge, replace each pixel by 4 copies.
Image Processing: Scaling Filter

Goal. Shrink or enlarge an image to desired size.

Uniform strategy. To convert from $w_s$-by-$h_s$ to $w_t$-by-$h_t$:

- Scale column index by $w_s / w_t$.
- Scale row index by $h_s / h_t$.
- Set color of pixel $(x, y)$ in target image to color of pixel $(x \times w_s / w_t, y \times h_s / h_t)$ in source image.

```
import java.awt.Color;
public class Scale {
    public static void main(String[] args) {
        String filename = args[0];
        int w = Integer.parseInt(args[1]);
        int h = Integer.parseInt(args[2]);
        Picture source = new Picture(filename);
        Picture target = new Picture(w, h);
        for (int tx = 0; tx < target.width(); tx++) {
            for (int ty = 0; ty < target.height(); ty++) {
                int sx = tx * source.width() / target.width();
                int sy = ty * source.height() / target.height();
                Color color = source.get(sx, sy);
                target.set(tx, ty, color);
            }
        }
        source.show();
        target.show();
    }
}
```

More Image Processing Effects

```
wave filter  glass filter  Sobel edge detection
```

Text Processing

String data type. Basis for text processing. Set of values. Sequence of Unicode characters.

API:
```java
public class String {
    String(String s) { create string with the same value as s }
    int length()      // length of string
    char charAt(int i) // get character
    boolean substring(int l, int j) // substring
    boolean contains(String sub) // check whether contains sub
    boolean startsWith(String pre) // true if starts with pre
    boolean endsWith(String post) // true if ends with post
    int indexOf(String s) // index of first occurrence in this string
    int lastIndexOf(String s) // index of last occurrence of s
    String concat(String s) // append s to this
    String replaceAll(String a, String b) // replace all a by b
    String split(String delim) // strings between occurrences of delim
    boolean equals(String s) // true if equals
}
```

http://download.oracle.com/javase/6/docs/api/java/lang/String.html
Typical String Processing Code

```java
public static boolean isPalindrome(String s) {
    int left = 0, right = s.length() - 1;
    while (left < right) {
        if (s.charAt(left) != s.charAt(right))
            return false;
        left++;
        right--;
    }
    return true;
}
```

String query = args[0];
while (StdIN.hasNextLine())
    String s = StdIN.readLine();
    if (s.contains(query)) StdOut.println(s);

String s = StdIn.readString();
if (s.endsWith("http://")) StdOut.println(s);
```

Gene Finding: Algorithm

Algorithm. Scan left-to-right through genome.
- If start codon, then set beg to index i.
- If stop codon and substring is a multiple of 3
  - output gene
  - reset beg to -1

<table>
<thead>
<tr>
<th>i</th>
<th>codon</th>
<th>start</th>
<th>stop</th>
<th>beg</th>
<th>gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TAG</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATG</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TAG</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TAG</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>TAG</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>ATG</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>TAG</td>
<td>23</td>
<td></td>
<td></td>
<td>TCC</td>
</tr>
</tbody>
</table>

Gene Finding: Implementation

```java
public class GeneFind {
    public static void main(String[] args) {
        String start = args[0];
        String stop = args[1];
        String genome = StdIn.readAll();
        int beg = -1;
        for (int i = 0; i < genome.length(); i++) {
            String codon = genome.substring(i, i+3);
            if (codon.equals(start))
                beg = i;
            if (codon.equals(stop) && beg != -1)
                if (beg < genome.length() - 3)
                    { if (genome.substring(beg+1, i+1).length() % 3 == 0) {
                        StdOut.println(genome.substring(beg, i+1));
                        beg = -1;
                    }
                    }
        }
    }
}
```

In and Out

Possible memory representation of a string.
- genome = "ACCAGTTTACG";
- start = 0, stop = 14
- genome length = 15
- memory address 0: A
  1: C
  2: C
  3: A
  4: G
  5: T
  6: T
  7: T
  8: A
  9: C
  10: G
  11: C
  12: T
  13: G
  14: C
Bird’s Eye View (Revisited)

Non-Standard Input

Standard input: Read from terminal window.
Goal: Read from several different input streams.

In data type: Read text from stdin, a file, a web site, or network.

Ex: Are two text files identical?

```
public class Diff {
    public static void main(String[] arg) {
        In in0 = new In(arg[0]);
        In in1 = new In(arg[1]);
        String s = in0.readAll();
        String t = in1.readAll();
        StdOut.println(s.equals(t));
    }
}
```

Screen Scraping

Goal: Find current stock price of Google.

```
http://finance.yahoo.com/q?s=goog
```

```
public class StockQuote {
    public static void main(String[] arg) {
        String name = "http://finance.yahoo.com/q?s=" + arg[0];
        In in = new In(name + "&f=ts1");
        String input = in.readAll();
        int start = input.indexOf("Last Trade:" , 0);
        int from = input.indexOf("<b>", start);
        int to = input.indexOf("</b>", from);
        String price = input.substring(from + 3, to);
        StdOut.println(price);
    }
}
```

Day Trader

Add bells and whistles:

- Plot price in real-time.
- Notify user if price dips below a certain price.
- Embed logic to determine when to buy and sell.
- Automatically send buy and sell orders to trading firm.

Warning: Please, please use at your own financial risk.

OOP Summary

Object: Holds a data type value; variable name refers to object.

In Java, programs manipulate references to objects.

- Exception: primitive types, e.g., boolean, int, double.
- Reference types: String, Picture, Color, arrays, everything else.
- OOP purist: language should not have separate primitive types.

Bottom line: We wrote programs that manipulate colors, pictures, and strings.

Next time: We’ll write programs that manipulate our own abstractions.
Extra Slides

Color Separation

```
import java.awt.Color;

public class ColorSeparation {
    public static void main(String args[]) {
        Picture pic = new Picture(args[0]);
        int width = pic.width();
        int height = pic.height();
        Picture R = new Picture(width, height);
        Picture G = new Picture(width, height);
        Picture B = new Picture(width, height);
        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                Color c = pic.get(x, y);
                int r = c.red();
                int g = c.green();
                int b = c.blue();
                R.set(x, y, new Color(r, 0, 0));
                G.set(x, y, new Color(0, g, 0));
                B.set(x, y, new Color(0, 0, b));
            }
        }
        R.show();
        G.show();
        B.show();
    }
}
```

Color Separation java. Creates three Picture objects and windows.

Memory Management

Value types.
- Allocate memory when variable is declared.
- Can reclaim memory when variable goes out of scope.

Reference types.
- Allocate memory when object is created with new.
- Can reclaim memory when last reference goes out of scope.
- Significantly more challenging if several references to same object.

Garbage collector. System automatically reclaims memory; programmer relieved of tedious and error-prone activity.